**LAB 1: GETTING STARTED SHARED MEMORY MIMD AND DISTRIBUTED MEMORY MIMD**

**GOAL**

In this lab:

* + You start to work with the Heracles cluster by learning its architecture, getting access, and monitoring the cluster status.
  + You will gain an introduction and become familiar with two different parallel programming languages: openMP (Open Multi-Processing) and MPI (Message Passing Interface), which are used to design and implement programs on two different parallel platforms. We use the matrix multiplication programs as examples to introduce these subjects. You will observe that different tools (parallel languages with various capabilities) and design considerations are needed for different parallel platforms. Note that openMP and MPI will be covered in more detail in future labs. Read all the sections at the end of this document in order to answer the questions.

If you need help with the labs, contact Thoria or Manh by sending email [pdslab@ucdenver.edu](mailto:pdslab@ucdenver.edu) with subject **“Question Lab01”**

**SUBMISSION**

* + Provide answers for each question in **QUESTIONs** section. Place your answers in a word document (.docx) and name it as “Lab1\_LastName\_FirstName.docx.”
  + Submit your document on Canvas.
  + Show your own work.

**QUESTIONS**

1. (5pts) According to the Heracles architecture, write a short paragraph in your own words about your understanding of this machine. Follow steps in Section 1 to answer this question.
2. (5pts) Give few examples of applications that can benefit from this type of machine and what challenges come with this type of machine?
3. In this lab, you are given three versions of matrix multiplication programs:

**mm-seq.cpp**: *sequential C++ CPU Version.*

**mm-omp.cpp:** *OpenMP Version.*

**mm-mpi.cpp:** *MPI Version.*

a) (10pts) Understand the given source codes and explain the differences in computing and execution models of these programs (i.e. how these programs are designed and executed on parallel hardware? Which one uses distributed and shared memory?). To understand the codes, please get yourself used with basic syntax of OpenMP and MPI. The following materials are useful to learn these syntaxes:

* <https://computing.llnl.gov/tutorials/openMP/>
* <https://computing.llnl.gov/tutorials/mpi/#LLNL>
* <http://mpi.deino.net/mpi_functions/index.htm>

b) (5pts) For each version of matrix multiplication**,** run the program with matrix sizes according to the Table, then fill out the runtime in Table 1. Follow steps in Section 2 to compile and run the codes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1: Runtime (Heracles Machine)** | | | |
| **Matrix size** | **mm-seq** | **mm-omp-h** | **mm-mpi**  **(1 node x 24 processes)** |
| 192 |  |  |  |
| 1008 |  |  |  |
| 2016 |  |  |  |
| 4080 |  |  |  |

c) (5pts) Plot the chart for each program version in Table 1 using the runtime(x-axis is the matrix size; y-axis is the runtime).

d) (10pts) Calculate the speedup of parallel versions compared to its sequential version on CPU and fill out Table 2. According to the runtime and speedups, which codes performed better? Why?

|  |  |  |
| --- | --- | --- |
| **Table 2: Speedup** | | |
| **Matrix size** | **mm-omp-h** | **mm-mpi** |
| 192 |  |  |
| 1008 |  |  |
| 2016 |  |  |
| 4080 |  |  |

**SECTION 1: ACCESSING HERACLES CLUSTER**

**Request for PDS Lab Access**

You should already have received an email from [pdslab@ucdenver.edu](mailto:pdslab@ucdenver.edu) about your account information and instructions to login on Heracles. If you did not receive the email until now, please contact the TAs immediately for your account info.

**Login**

Once you obtained an account information, you will be able to login to Heracles with host name: **heracles.ucdenver.pvt.** Go to this link in order to learn how to connect to the Cluster

<http://pds.ucdenver.edu/document.php?type=cluster&name=access#Accessing_the_hardware>

If you are not on campus, remember to connect to the campus network via VPN before logging on Hereacles. More information about VPN and remote access is at: <https://www.ucdenver.edu/offices/office-of-information-technology/software/how-do-i-use/vpn-and-remote-access>

1. Learn HERACLES Multi-Core Cluster architecture:

<http://pds.ucdenver.edu/webclass/Heracles_Architecture.html>

1. Monitoring Heracles by accessing the following link in order to observe some metrics (CPU, Network, memory, etc.)
   1. Connect to the VPN if you are not on Campus before accessing the link.

[https://heracles.ucdenver.pvt/mcms/](https://heracles/mcms/)

* 1. if you get this error bellow, just click on **<Advanced>** and **<Proceed to Heracles(unsafe)>**

|  |
| --- |
|  |
|  | | |

**SECTION 2: COMPILING PROGRAMS ON HERACLES**

If you are not familiar with Linux, you should get started with any online Linux tutorial before doing this and future Lab assignments. Some Linux tutorials are available at: <http://pds.ucdenver.edu/document.php?type=links&name=tutorials>

1. **Copy source code files to Heracles**

* Logon to the cluster
* Make a folder named csc5551/lab1 in your home directory using the ‘mkdir’ command.

$ mkdir /path/to/directory

For example:

$ mkdir /home/john/csc5551/lab01

* Copy the source code files to your folder. If you are using MAC or Linux, you can copy these files using scp or sftp command. Also you can download Cyberduck for Mac (<http://download.cnet.com/Cyberduck/3000-2160_4-10246246.html> )

If you are using Windows, you can use WinSCP or SSH Secure Shell to login and copy files from your PC to the cluster. For more information on downloading WinSCP and how to use this software, visit:

<http://pds.ucdenver.edu/document.php?type=software&name=winscp>

* Change the working directory to csc5551/csc7551 using cd command

$ cd /path/to/directory

For example:

$ cd /home/john/csc5551/labnn

1. **Compiling programs.**

Update intel compiler library, execute the following command every time you log on, or you can insert in in ~/.bashrc for an automatic update.

$ source /opt/intel/compilers\_and\_libraries\_2019.4.243/linux/bin/compilervars.sh -arch intel64 -platform linux

Use the following commands to compile given source codes. Make sure the executable files are: mm-seq, mm-omp, mm-mpi for the slurm scripts to work. Otherwise, you need to modify the slurm scripts.

$ g++ -O mm-seq.cpp -o mm-seq

$ g++ -O -fopenmp mm-omp.cpp -o mm-omp

$ mpicxx mm-mpi.cpp -o mm-mpi

**For more detail instructions, please read:**

* Compiling sequential programs on Heracles (C, C++ and Fortran)

<http://pds.ucdenver.edu/webclass/Compiling%20C_C++%20and%20Fortran%20programs.html>

* Compiling MPI code on HERACLES

<http://pds.ucdenver.edu/webclass/Compiling%20MPI%20program%20on%20Heracles.html>

* Compiling OpenMP programs on Heracles

<http://pds.ucdenver.edu/webclass/Compiling%20openMP%20programs.html>

* Compiling SIMD-SSE programs <http://pds.ucdenver.edu/webclass/Compiling%20openMP_SSE%20programs%20on%20Hydra.html>
* Compiling CUDA programs on Heracles

<http://pds.ucdenver.edu/webclass/Heracles-Compiling%20Cuda%20code.html>

**SECTION 3: RUNNING PROGRAMS ON HERACLES**

We use Slurm to run the compiled codes (generated from Section 2) as it will guarantee that the codes will run on an available node(s) without any other concurrent jobs. It is crucial for accurate runtime measurements. In this lab, we have provided the Slurm script for each given source code as follows:

* c\_slurm.sh (script to run C/C++ sequential code)
* openmp\_slurm.sh(script to run openMP code)
* mpi\_slurm.sh(script to run MPI code)

**Run Slurm scripts.**

From the master node, execute the following command:

$ sbatch scriptName argument1 argument2

where,

* scriptName is either **c\_slurm.sh** or **openmp\_slurm.sh** or **mpi\_slurm.sh** depending on which program you want to execute.
* argument1 is the dimension of the matrix
* argument2: is the option to specify if input/output matrices are printed or not (1=print; 0= not print). Program only prints if the matrix size is less than 10.

By default, output results will be written to file **slurm\_output.jobID** and errors will be written to file **slurm\_error.jobID,** where **jobID** is a job id number, which is assigned when you execute the command sbatch.

For example:

$ sbatch c\_slurm.sh 4 1

will run mm-seq (executable file of C/C++ sequential version) with matrix size 4x4 and print out results.

**Useful commands with Slurm scripts:**

squeue // check the job queue

scancel jobID // cancel job execution

**3 (Optional) Configuring Slurm.**

You may change some parameters in your script, such as:

#SBATCH --mail-type=ALL ### configure elamil

#SBATCH --mail-user=myemailaddress ### put your email

#SBATCH --job-name=myjob ### Job Name

#SBATCH --output=slurm\_output.%j ### File in which to store job output

#SBATCH --error=slurm\_error.%j ### File in which to store job error messages

For MPI job you may configure the grid by changing some parameters in your job according to the experiment, such as:

#SBATCH --ntasks=24 ## corresponds to MPI ranks/ each rank corresponds to one task

#SBATCH --ntasks-per-node=24 ## Number of tasks to be launched per Node, default = 1

In the above example the mpi job will launch 24 tasks divided in 24 tasks per node, in other word the MPI job will use one node run 24 tasks.

Check this link for more information about Slurm scripts:

<http://pds.ucdenver.edu/webclass/Heracles-RunningPrograms%20Slurm.html>